4. b) O alta echipa de programatori a dezvoltat algoritmul de lock prezentat in pseudocodul urmator ce incapsuleaza un alt lock oarecare. Se considera ca lock-ul incapsulat asigura corect excluderea mutuala si este starvation-free. De asemenea lock-ul incapsulat permite un apel unlock fara exceptie si fara efect chiar daca nu a existat un apel lock. ThreadId se considera a fi o clasa ce furnizeaza un id unic pozitiv fiecarui thread.

Intr-o executie concurenta a n > 1 thread-uri, asigura acest algoritm excluderea mutuala? Argumentati.

```
1 class VeryShadyLock {
2    private Lock lock;
3    private volatile int x, y = 0;
4
5    public void lock() {
6       int me = ThreadId.get();
7       x = me;
8       while (y != 0) {};
9       y = me;
10      if (x != me) {
11         lock.lock();
12      }
13   }
14
15   public void unlock() {
16      y = 0;
17      lock.unlock();
18   }
19 }
```

Excluderea mutuala asigura ca, in orice moment, doar un singur thread poate accesa sectiunea critica.

In clasa VeryShadyLock, variabila volatila x (vizibila tuturor threadurilor) este utilizata pentru a indica threadul care doreste acces la sectiunea critica. De asemenea, variabila volatila y serveste ca indicator al starii sectiunii critice: cand y = 0, sectiunea critica este libera; cand y ≠ 0,

sectiunea critica este ocupata, iar valoarea lui y reprezinta ID-ul threadului care are in acel moment acces la sectiunea critica.

Algoritmul nu asigura excluderea mutuala, deoarece doua thread-uri pot avea acces la sectiunea critica simultan.

Trace demonstrativ:

1. Doua fire de executie T1 si T2 intra in functia lock aproape simultan: T1 seteaza variabila x = 1; si T2 o suprascrie la x = 2.
2. T1 si T2 intra in bucla *while (y != 0) {};* si asteapta eliberarea sectiunii critice.
3. Sectiunea critica se elibereaza si y devine 0, ambele threaduri seteaza *y = me;*
    - T1 seteaza variabila y = 1;
    - T2 o suprascrie la y =2;
4. T1 si T2 verifica *if (x != me)*:
    - T1 trece de aceasta conditie (*if(2 != 1)*) si apeleaza *lock.lock();*
    - T2 nu trece de aceasta conditie (*if(2 != 2)*), deci nu va apela lock-ul si va intra direct in sectiunea critica, simultan cu T1, ceea ce incalca excluderea mutuala.